Assinatura Digital

Prof. Dr. Iaçanã Ianiski Weber

Confiabilidade e Segurança de Software

98G08-4

Agradecimentos especiais ao Prof. Avelino Zorzo e aos Autores Christof Paar e Jan Pelzl pelo material.

Índice

Índice

Motivação

- Alice encomenda um carro rosa com bancos alaranjados do vendedor Bob.
- Após ver que o carro rosa não combinou com os bancos alaranjados, Alice alega que nunca o encomendou.
- Como Bob pode provar a um juiz que Alice realmente encomendou o carro? (E que ele mesmo não forjou o pedido?)

Motivação

- Alice encomenda um carro rosa com bancos alaranjados do vendedor Bob.
- Após ver que o carro rosa não combinou com os bancos alaranjados, Alice alega que nunca o encomendou.
- Como Bob pode provar a um juiz que Alice realmente encomendou o carro? (E que ele mesmo não forjou o pedido?)

$\Rightarrow$ A criptografia simétrica falha porque tanto Alice quanto Bob podem ser maliciosos.

$\Rightarrow$ O problema pode ser resolvido com criptografia de chave pública.

Motivação

- Alice encomenda um carro rosa com bancos alaranjados do vendedor Bob.
- Após ver que o carro rosa não combinou com os bancos alaranjados, Alice alega que nunca o encomendou.
- Como Bob pode provar a um juiz que Alice realmente encomendou o carro? (E que ele mesmo não forjou o pedido?)

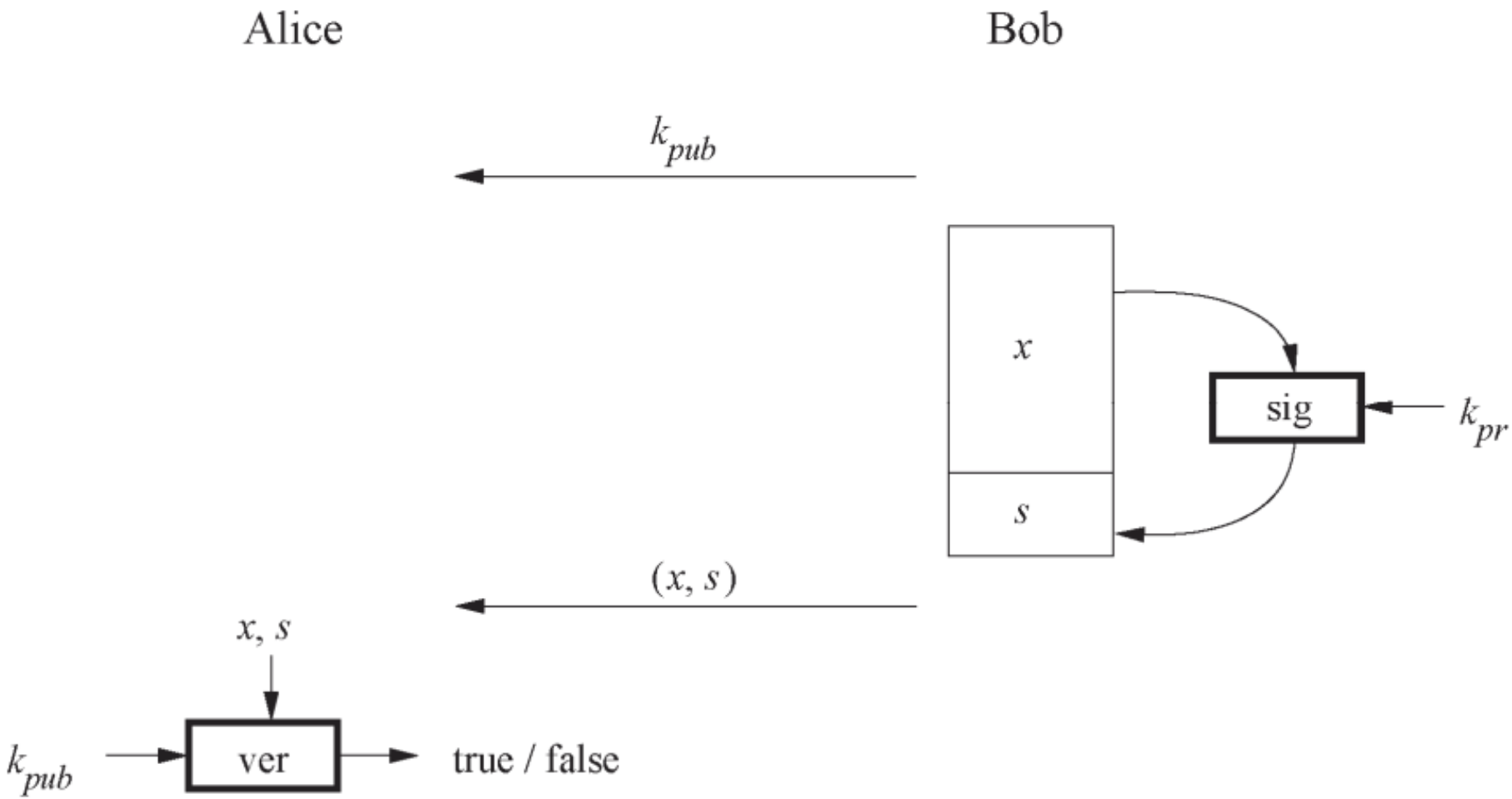
$\Rightarrow$ A criptografia simétrica falha porque tanto Alice quanto Bob podem ser maliciosos.

$\Rightarrow$ O problema pode ser resolvido com criptografia de chave pública.

Conceito Central

Este cenário ilustra a necessidade de uma propriedade de segurança chamada **não repúdio** (ou irretratabilidade).

Princípio Básico da Assinatura Digital



Ideia Principal

- Para uma dada mensagem x, uma assinatura digital é anexada à mensagem (de forma análoga a uma assinatura convencional).
- Apenas a pessoa com a *private key* (chave privada) deve ser capaz de gerar a assinatura.
- A assinatura deve ser diferente para cada documento.

$\Rightarrow$ A assinatura é gerada por uma função que tem como entradas a mensagem x e a *private key*.

$\Rightarrow$ A *public key* (chave pública) e a mensagem x são as entradas para a função de verificação.

Índice

Serviços Essenciais de Segurança

Os objetivos de um sistema de segurança são chamados de *serviços de segurança*.

1. **Confidencialidade:** Garante que a informação seja acessível apenas por partes autorizadas.

Serviços Essenciais de Segurança

Os objetivos de um sistema de segurança são chamados de *serviços de segurança*.

1. **Confidencialidade:** Garante que a informação seja acessível apenas por partes autorizadas.
2. **Integridade:** Garante que a mensagem não foi modificada em trânsito.

Serviços Essenciais de Segurança

Os objetivos de um sistema de segurança são chamados de *serviços de segurança*.

1. **Confidencialidade:** Garante que a informação seja acessível apenas por partes autorizadas.
2. **Integridade:** Garante que a mensagem não foi modificada em trânsito.
3. **Autenticação de Mensagem:** Garante que o remetente de uma mensagem é autêntico. Um termo alternativo é *autenticação de origem de dados*.

Serviços Essenciais de Segurança

Os objetivos de um sistema de segurança são chamados de *serviços de segurança*.

1. **Confidencialidade:** Garante que a informação seja acessível apenas por partes autorizadas.
2. **Integridade:** Garante que a mensagem não foi modificada em trânsito.
3. **Autenticação de Mensagem:** Garante que o remetente de uma mensagem é autêntico. Um termo alternativo é *autenticação de origem de dados*.
4. **Não Repúdio (ou Irretratabilidade):** Garante que o remetente de uma mensagem não possa negar a autoria da mesma (vide o exemplo do carro rosa).

Serviços de Segurança Adicionais

5. **Identificação/Autenticação de Entidade:** Estabelecimento e verificação da identidade de uma entidade (ex: uma pessoa, um computador, um cartão de crédito).

Serviços de Segurança Adicionais

5. **Identificação/Autenticação de Entidade:** Estabelecimento e verificação da identidade de uma entidade (ex: uma pessoa, um computador, um cartão de crédito).
6. **Controle de Acesso:** Restrição do acesso a recursos apenas para entidades privilegiadas.

Serviços de Segurança Adicionais

5. **Identificação/Autenticação de Entidade:** Estabelecimento e verificação da identidade de uma entidade (ex: uma pessoa, um computador, um cartão de crédito).
6. **Controle de Acesso:** Restrição do acesso a recursos apenas para entidades privilegiadas.
7. **Disponibilidade:** Garantia de que o sistema eletrônico esteja disponível de forma confiável.

Serviços de Segurança Adicionais

5. **Identificação/Autenticação de Entidade:** Estabelecimento e verificação da identidade de uma entidade (ex: uma pessoa, um computador, um cartão de crédito).
6. **Controle de Acesso:** Restrição do acesso a recursos apenas para entidades privilegiadas.
7. **Disponibilidade:** Garantia de que o sistema eletrônico esteja disponível de forma confiável.
8. **Auditoria:** Fornecimento de evidências sobre atividades relevantes para a segurança, por exemplo, mantendo *logs* sobre certos eventos.

Serviços de Segurança Adicionais

5. **Identificação/Autenticação de Entidade:** Estabelecimento e verificação da identidade de uma entidade (ex: uma pessoa, um computador, um cartão de crédito).
6. **Controle de Acesso:** Restrição do acesso a recursos apenas para entidades privilegiadas.
7. **Disponibilidade:** Garantia de que o sistema eletrônico esteja disponível de forma confiável.
8. **Auditoria:** Fornecimento de evidências sobre atividades relevantes para a segurança, por exemplo, mantendo *logs* sobre certos eventos.
9. **Segurança Física:** Proteção contra adulteração física e/ou respostas a tentativas de adulteração física.

Serviços de Segurança Adicionais

5. **Identificação/Autenticação de Entidade:** Estabelecimento e verificação da identidade de uma entidade (ex: uma pessoa, um computador, um cartão de crédito).
6. **Controle de Acesso:** Restrição do acesso a recursos apenas para entidades privilegiadas.
7. **Disponibilidade:** Garantia de que o sistema eletrônico esteja disponível de forma confiável.
8. **Auditoria:** Fornecimento de evidências sobre atividades relevantes para a segurança, por exemplo, mantendo *logs* sobre certos eventos.
9. **Segurança Física:** Proteção contra adulteração física e/ou respostas a tentativas de adulteração física.
10. **Anonimato:** Proteção contra a descoberta e o uso indevido da identidade.

Índice

Ideia Principal do Esquema de Assinatura RSA

Para gerar a private key e a public key:

- Usa-se o mesmo processo de geração de chaves da criptografia RSA.

Ideia Principal do Esquema de Assinatura RSA

Para gerar a private key e a public key:

- Usa-se o mesmo processo de geração de chaves da criptografia RSA.

Para gerar a assinatura:

- "Criptografa-se" a mensagem x com a *private key* (d, n).

$$s = \text{sig}_{K_{priv}}(x) = x^d \pmod{n}$$

- Anexa-se a assinatura s à mensagem x.

Ideia Principal do Esquema de Assinatura RSA

Para gerar a private key e a public key:

- Usa-se o mesmo processo de geração de chaves da criptografia RSA.

Para gerar a assinatura:

- "Criptografa-se" a mensagem x com a *private key* (d, n).

$$s = \text{sig}_{K_{priv}}(x) = x^d \pmod{n}$$

- Anexa-se a assinatura s à mensagem x.

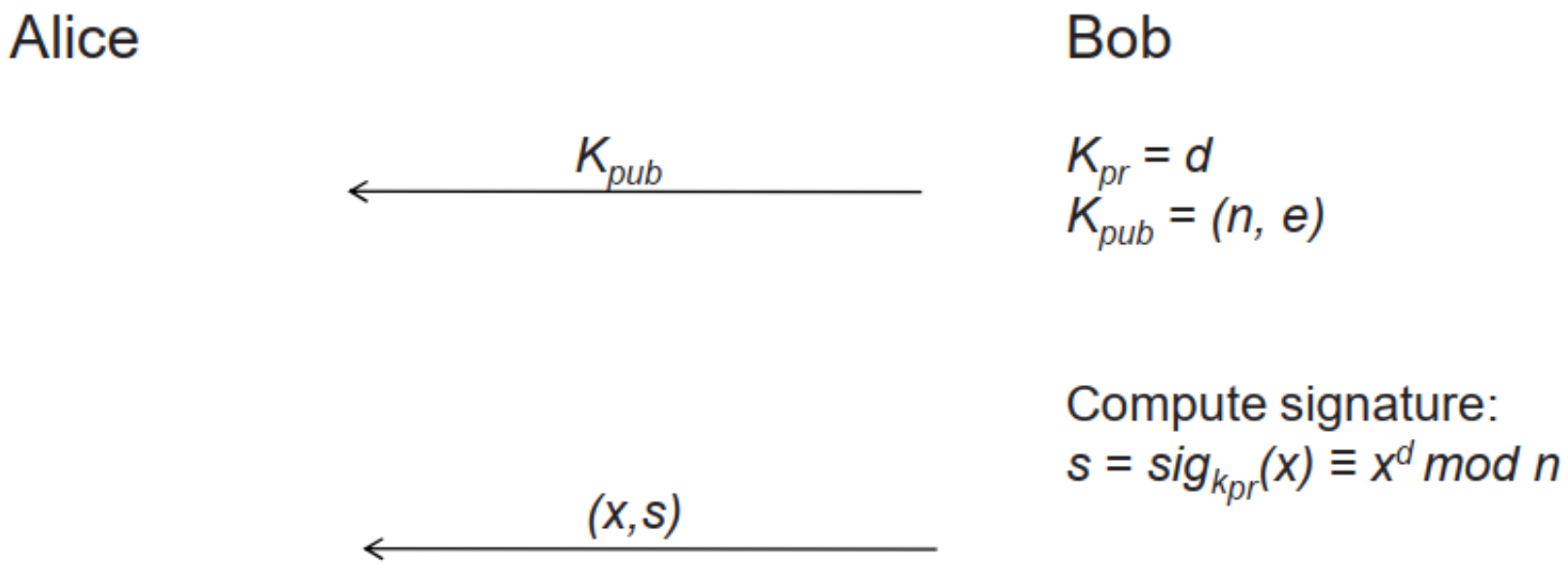
Para verificar a assinatura:

- "Decriptografa-se" a assinatura s com a *public key* (e, n).

$$x' = \text{ver}_{K_{pub}}(s) = s^e \pmod{n}$$

- Se $x = x'$, a assinatura é válida.

O Protocolo de Assinatura RSA



Segurança e Desempenho do Esquema de Assinatura RSA

Segurança (Recomendações Atuais do NIST):

- A segurança do RSA depende do tamanho em bits do módulo n. As recomendações do NIST (SP 800-57) são:
  - **Mínimo Aceitável:** Chave de **2048 bits** para um nível de segurança de 112 bits.
  - **Padrão Recomendado:** Chave de **3072 bits** para um nível de segurança de 128 bits.

Segurança e Desempenho do Esquema de Assinatura RSA

Segurança (Recomendações Atuais do NIST):

- A segurança do RSA depende do tamanho em bits do módulo n. As recomendações do NIST (SP 800-57) são:
  - **Mínimo Aceitável:** Chave de **2048 bits** para um nível de segurança de 112 bits.
  - **Padrão Recomendado:** Chave de **3072 bits** para um nível de segurança de 128 bits.

$\Rightarrow$ Consequentemente, a assinatura s terá o mesmo tamanho do módulo n (por exemplo, 2048 ou 3072 bits).

Segurança e Desempenho do Esquema de Assinatura RSA

Segurança (Recomendações Atuais do NIST):

- A segurança do RSA depende do tamanho em bits do módulo n. As recomendações do NIST (SP 800-57) são:
  - **Mínimo Aceitável:** Chave de **2048 bits** para um nível de segurança de 112 bits.
  - **Padrão Recomendado:** Chave de **3072 bits** para um nível de segurança de 128 bits.

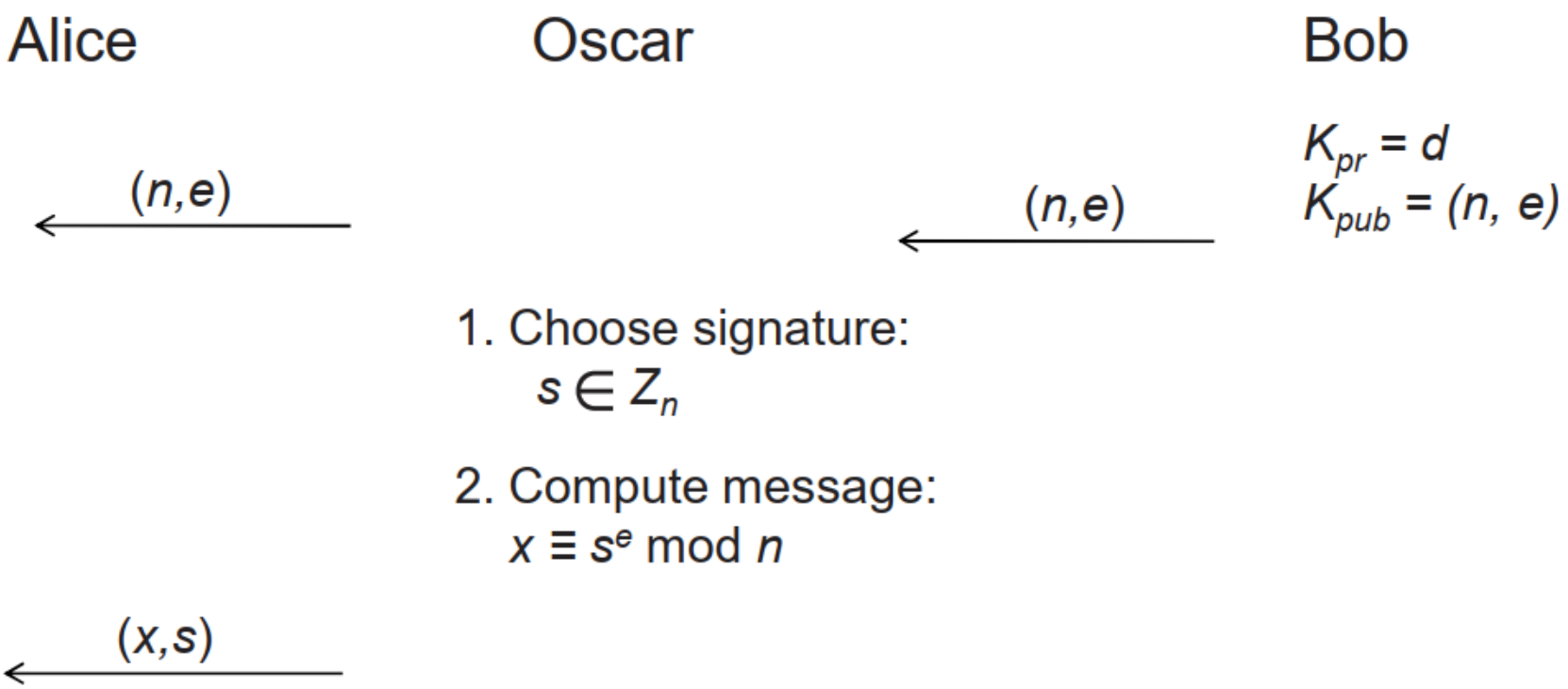
$\Rightarrow$ Consequentemente, a assinatura s terá o mesmo tamanho do módulo n (por exemplo, 2048 ou 3072 bits).

Desempenho:

- O processo de assinatura é uma exponenciação com a *private key* (d), e a verificação é uma exponenciação com a *public key* (e).

$\Rightarrow$ A verificação da assinatura é muito eficiente, pois um número pequeno pode ser escolhido para a *public key* (um valor comum é $e = 65537$).

Ataque de Falsificação Existencial contra a Assinatura RSA



Verification:
$s^e \equiv x' \bmod n$

since $s^e = (x^d)^e \equiv x \bmod n$
$\rightarrow$ Signature is valid

Falsificação Existencial e Padding

- Um atacante pode gerar pares de mensagem-assinatura (x, s) que são matematicamente válidos.
- Contudo, o atacante só pode escolher a assinatura s, e **NÃO** a mensagem x que resulta dela.
  - $\Rightarrow$ Assim, o atacante não consegue forjar uma assinatura para uma mensagem de sua escolha (ex: *"Transferir $1000 para a conta de Oscar"*).

Falsificação Existencial e Padding

- Um atacante pode gerar pares de mensagem-assinatura (x, s) que são matematicamente válidos.
- Contudo, o atacante só pode escolher a assinatura s, e **NÃO** a mensagem x que resulta dela.
  - $\Rightarrow$ Assim, o atacante não consegue forjar uma assinatura para uma mensagem de sua escolha (ex: *"Transferir $1000 para a conta de Oscar"*).

Solução: Esquemas de Padding

Formatar a mensagem x de acordo com um **esquema de padding** (como o PKCS#1 v1.5 ou PSS) antes de assinar é a contramedida padrão para este ataque.

Uma mensagem x gerada pelo atacante a partir de uma assinatura aleatória s terá um formato igualmente aleatório. A probabilidade de que este formato coincida com a estrutura exigida pelo esquema de padding é computacionalmente desprezível.

O Padrão Clássico: PKCS#1 v1.5

Este foi o primeiro padrão amplamente adotado para formatação de mensagens antes da assinatura RSA. Seu objetivo é criar um bloco de dados com uma estrutura fixa e previsível.

Estrutura do Bloco a ser Assinado:

`00 01 FF FF ... FF 00 [ASN.1 HASH_ID] [HASH(M)]`

- `00 01`: Bytes fixos que indicam o modo de assinatura.
- `FF...FF`: Uma série de bytes '0xFF' (padding) para preencher o espaço até o tamanho do módulo RSA (ex: 2048 bits).
- `00`: Um byte separador.
- `[ASN.1 HASH_ID]`: Um identificador fixo para o algoritmo de hash usado (ex: SHA-256). Previne ataques onde o algoritmo de hash poderia ser trocado.
- `[HASH(M)]`: O hash da mensagem original.

Característica Principal:

O processo é **determinístico**: a mesma mensagem sempre gera o mesmo bloco formatado. É funcional e amplamente compatível, mas considerado legado para novas aplicações.

O Padrão Moderno e Recomendado: PSS

O **P**robabilistic **S**ignature **S**cheme (PSS) foi projetado para ser mais robusto e possui uma prova de segurança formal, sendo a recomendação para todas as novas aplicações.

Ideia Central: Introduzir Aleatoriedade O PSS utiliza dois componentes chave para tornar o padding não-determinístico:

- **Sal (Salt):** Um valor aleatório gerado para cada nova assinatura. Garante que assinar a mesma mensagem duas vezes produza resultados completamente diferentes.
- **MGF (Mask Generation Function):** Uma função (baseada em hash) que "embaralha" os dados, criando um bloco de aparência aleatória e escondendo a estrutura interna.

Processo Simplificado: `MGF(HASH(M) | RNG Salt)` $\rightarrow$ **Bloco Final**

Característica Principal

O processo é **probabilístico**: graças ao "sal" aleatório, a mesma mensagem gera um bloco formatado diferente a cada assinatura. Isso elimina diversas vulnerabilidades teóricas.

Índice

Fatos sobre o Digital Signature Algorithm (DSA)

- É um padrão do Governo Federal dos EUA para assinaturas digitais, parte do **Digital Signature Standard (DSS)**.
  - *Nota: O padrão atual (FIPS 186-5)* **descontinuou** *a geração de novas assinaturas com DSA após 2023. A verificação ainda é permitida para legado.*

Fatos sobre o Digital Signature Algorithm (DSA)

- É um padrão do Governo Federal dos EUA para assinaturas digitais, parte do **Digital Signature Standard (DSS)**.
  - *Nota: O padrão atual (FIPS 186-5)* **descontinuou** *a geração de novas assinaturas com DSA após 2023. A verificação ainda é permitida para legado.*
- Foi proposto pelo NIST (*National Institute of Standards and Technology*).

Fatos sobre o Digital Signature Algorithm (DSA)

- É um padrão do Governo Federal dos EUA para assinaturas digitais, parte do **Digital Signature Standard (DSS)**.
  - *Nota: O padrão atual (FIPS 186-5)* **descontinuou** *a geração de novas assinaturas com DSA após 2023. A verificação ainda é permitida para legado.*
- Foi proposto pelo NIST (*National Institute of Standards and Technology*).
- É baseado no esquema de assinatura ElGamal, que por sua vez depende da dificuldade do problema do logaritmo discreto.

Fatos sobre o Digital Signature Algorithm (DSA)

- É um padrão do Governo Federal dos EUA para assinaturas digitais, parte do **Digital Signature Standard (DSS)**.
  - *Nota: O padrão atual (FIPS 186-5)* **descontinuou** *a geração de novas assinaturas com DSA após 2023. A verificação ainda é permitida para legado.*
- Foi proposto pelo NIST (*National Institute of Standards and Technology*).
- É baseado no esquema de assinatura ElGamal, que por sua vez depende da dificuldade do problema do logaritmo discreto.
- O tamanho da assinatura era de 320 bits em versões mais antigas do padrão.
  - No padrão atual, os tamanhos são de **448 ou 512 bits**.

Fatos sobre o Digital Signature Algorithm (DSA)

- É um padrão do Governo Federal dos EUA para assinaturas digitais, parte do **Digital Signature Standard (DSS)**.
  - *Nota: O padrão atual (FIPS 186-5)* **descontinuou** *a geração de novas assinaturas com DSA após 2023. A verificação ainda é permitida para legado.*
- Foi proposto pelo NIST (*National Institute of Standards and Technology*).
- É baseado no esquema de assinatura ElGamal, que por sua vez depende da dificuldade do problema do logaritmo discreto.
- O tamanho da assinatura era de 320 bits em versões mais antigas do padrão.
  - No padrão atual, os tamanhos são de **448 ou 512 bits**.
- A verificação da assinatura é geralmente mais lenta que a do RSA (especialmente quando o RSA usa um expoente público pequeno, como $e = 65537$).

Lições Aprendidas e Conclusões

- Assinaturas digitais fornecem três serviços de segurança essenciais: **integridade**, **autenticação de mensagem** e **não repúdio**.

Lições Aprendidas e Conclusões

- Assinaturas digitais fornecem três serviços de segurança essenciais: **integridade**, **autenticação de mensagem** e **não repúdio**.
- Embora o RSA ainda seja amplamente utilizado, o **ECDSA (Elliptic Curve DSA)** tornou-se o padrão para a maioria das novas aplicações (web, blockchain, etc.) por oferecer a mesma segurança com chaves muito menores e mais eficientes.

Lições Aprendidas e Conclusões

- Assinaturas digitais fornecem três serviços de segurança essenciais: **integridade**, **autenticação de mensagem** e **não repúdio**.
- Embora o RSA ainda seja amplamente utilizado, o **ECDSA (Elliptic Curve DSA)** tornou-se o padrão para a maioria das novas aplicações (web, blockchain, etc.) por oferecer a mesma segurança com chaves muito menores e mais eficientes.
- Os principais algoritmos de assinatura hoje são:
  - **ECDSA**: O mais estabelecido e popular atualmente.
  - **EdDSA** (ex: Ed25519): Um algoritmo mais moderno, projetado para ser seguro, rápido e mais simples de implementar corretamente.
  - **DSA**: Um algoritmo legado, já descontinuado pelo NIST para novas implementações.

Lições Aprendidas e Conclusões

- Assinaturas digitais fornecem três serviços de segurança essenciais: **integridade**, **autenticação de mensagem** e **não repúdio**.
- Embora o RSA ainda seja amplamente utilizado, o **ECDSA (Elliptic Curve DSA)** tornou-se o padrão para a maioria das novas aplicações (web, blockchain, etc.) por oferecer a mesma segurança com chaves muito menores e mais eficientes.
- Os principais algoritmos de assinatura hoje são:
  - **ECDSA**: O mais estabelecido e popular atualmente.
  - **EdDSA** (ex: Ed25519): Um algoritmo mais moderno, projetado para ser seguro, rápido e mais simples de implementar corretamente.
  - **DSA**: Um algoritmo legado, já descontinuado pelo NIST para novas implementações.
- No RSA, a **verificação** é computacionalmente mais rápida que a **assinatura**, pois o expoente público e pode ser um número pequeno.

Índice

Introdução à Criptografia Pós-Quântica (PQC)

- Os algoritmos que usamos hoje (RSA, ECDSA, DSA) baseiam-se em problemas matemáticos que seriam facilmente resolvidos por um computador quântico em larga escala.
  - **Algoritmo de Shor:** Quebra o RSA e o ECDSA ao resolver a fatoração de inteiros e o logaritmo discreto de forma eficiente.

- A **Criptografia Pós-Quântica (PQC)** desenvolve novos algoritmos resistentes a ataques de computadores clássicos **e** quânticos.

- Em 2022, o NIST selecionou os primeiros algoritmos para padronização, incluindo:
  - **ML-KEM (CRYSTALS-Kyber):** Para troca de chaves.
  - **ML-DSA (CRYSTALS-Dilithium):** Para assinaturas digitais.

O Lema da PQC

Migre hoje, para não ter seus dados decifrados amanhã. (Harvest now, decrypt later)

O que é o ML-DSA (CRYSTALS-Dilithium)?

- **ML-DSA** significa *Module-Lattice-based Digital Signature Algorithm*.

- É o algoritmo de assinatura digital selecionado pelo NIST e padronizado no documento **FIPS 204**.

- Sua segurança não se baseia em fatoração ou logaritmos, mas sim na dificuldade de problemas em estruturas matemáticas chamadas **reticulados sobre módulos** (*module lattices*).
  - Problemas como o *Shortest Vector Problem* (SVP) e *Learning With Errors* (LWE) são considerados difíceis até mesmo para computadores quânticos.

- A ideia central é que o signatário prova conhecer uma informação secreta (um conjunto de vetores "curtos" em um reticulado), sem revelar essa informação.

Como o ML-DSA Funciona (Visão Geral)

1. **Geração de Chaves:**
   - Gera-se uma grande matriz pública $\mathbf{A}$ e dois vetores secretos e "curtos" $\mathbf{s}_1, \mathbf{s}_2$ (a chave privada).
   - A chave pública é a matriz $\mathbf{A}$ e o vetor $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$.

Como o ML-DSA Funciona (Visão Geral)

1. **Geração de Chaves:**
   - Gera-se uma grande matriz pública $\mathbf{A}$ e dois vetores secretos e "curtos" $\mathbf{s}_1, \mathbf{s}_2$ (a chave privada).
   - A chave pública é a matriz $\mathbf{A}$ e o vetor $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$.
2. **Assinatura:**
   - Usa uma técnica chamada *Fiat-Shamir with Aborts*.
   - O signatário gera um vetor aleatório ("máscara"), cria um "compromisso" público, e então usa um hash da mensagem e do compromisso para construir uma resposta.
   - A resposta (a assinatura) esconde a chave privada $\mathbf{s}_1, \mathbf{s}_2$, mas prova que o signatário a conhece. O processo pode "abortar" e recomeçar para evitar qualquer vazamento de informação.

Como o ML-DSA Funciona (Visão Geral)

1. **Geração de Chaves:**
   - Gera-se uma grande matriz pública $\mathbf{A}$ e dois vetores secretos e "curtos" $\mathbf{s}_1, \mathbf{s}_2$ (a chave privada).
   - A chave pública é a matriz $\mathbf{A}$ e o vetor $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$.
2. **Assinatura:**
   - Usa uma técnica chamada *Fiat-Shamir with Aborts*.
   - O signatário gera um vetor aleatório ("máscara"), cria um "compromisso" público, e então usa um hash da mensagem e do compromisso para construir uma resposta.
   - A resposta (a assinatura) esconde a chave privada $\mathbf{s}_1, \mathbf{s}_2$, mas prova que o signatário a conhece. O processo pode "abortar" e recomeçar para evitar qualquer vazamento de informação.
3. **Verificação:**
   - O verificador usa a chave pública $(\mathbf{A}, \mathbf{t})$ e a assinatura para recalcular o "compromisso".
   - Se o valor recalculado corresponder ao que está na assinatura e a assinatura for "curta" o suficiente, ela é considerada válida.

ML-DSA vs. Padrões Atuais: Trade-offs

Comparação aproximada para nível de segurança de 128 bits.

Parâmetro	**RSA-3072**	**ECDSA (P-256)**	**ML-DSA-2**
Segurança Quântica	Não	Não	Sim
Tam. Chave Pública	$\approx$ 384 bytes	$\approx$ 64 bytes	$\approx$ 1312 bytes
Tam. da Assinatura	$\approx$ 384 bytes	$\approx$ 64 bytes	$\approx$ 2420 bytes
Velocidade (Assinar)	Lenta	Rápida	**Muito Rápida**
Velocidade (Verificar)	**Muito Rápida**	Rápida	Rápida

Principal Conclusão

ML-DSA oferece segurança quântica ao custo de chaves e assinaturas consideravelmente maiores que as do ECDSA, mas com excelente performance de assinatura.

Parte 1: Geração de Chaves

1. Parâmetros Públicos

Definimos um módulo primo $q = 17$ e uma matriz pública $\mathbf{A}$:

$$\mathbf{A} = \begin{pmatrix} 10 & 3 \\ 5 & 9 \end{pmatrix}$$

(Nota: No ML-DSA real, estes seriam objetos matemáticos mais complexos, como polinômios.)

2. Geração de Chaves (Bob)

Bob gera sua chave privada, um vetor $\mathbf{s}$ com componentes "pequenos":

$$\mathbf{s} = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$$

Parte 1: Geração de Chaves

1. Parâmetros Públicos

Definimos um módulo primo $q = 17$ e uma matriz pública $\mathbf{A}$:

$$\mathbf{A} = \begin{pmatrix} 10 & 3 \\ 5 & 9 \end{pmatrix}$$

(Nota: No ML-DSA real, estes seriam objetos matemáticos mais complexos, como polinômios.)

2. Geração de Chaves (Bob)

Bob gera sua chave privada, um vetor $\mathbf{s}$ com componentes "pequenos":

$$\mathbf{s} = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$$

Ele então calcula o vetor $\mathbf{t} = \mathbf{As} \pmod{q}$ para obter o restante de sua chave pública:

$$\mathbf{t} = \begin{pmatrix} 10 & 3 \\ 5 & 9 \end{pmatrix} \begin{pmatrix} 2 \\ -1 \end{pmatrix} = \begin{pmatrix} 17 \\ 1 \end{pmatrix} \equiv \begin{pmatrix} 0 \\ 1 \end{pmatrix} \pmod{17}$$

Parte 1: Geração de Chaves

1. Parâmetros Públicos

Definimos um módulo primo $q = 17$ e uma matriz pública $\mathbf{A}$:

$$\mathbf{A} = \begin{pmatrix} 10 & 3 \\ 5 & 9 \end{pmatrix}$$

(Nota: No ML-DSA real, estes seriam objetos matemáticos mais complexos, como polinômios.)

Chaves Finais:

- **Pública:** $(\mathbf{A}, \mathbf{t})$
- **Privada:** $\mathbf{s}$

2. Geração de Chaves (Bob)

Bob gera sua chave privada, um vetor $\mathbf{s}$ com componentes "pequenos":

$$\mathbf{s} = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$$

Ele então calcula o vetor $\mathbf{t} = \mathbf{As} \pmod q$ para obter o restante de sua chave pública:

$$\mathbf{t} = \begin{pmatrix} 10 & 3 \\ 5 & 9 \end{pmatrix} \begin{pmatrix} 2 \\ -1 \end{pmatrix} = \begin{pmatrix} 17 \\ 1 \end{pmatrix} \equiv \begin{pmatrix} 0 \\ 1 \end{pmatrix} \pmod{17}$$

Parte 2: Geração da Assinatura

Bob quer assinar uma mensagem M. Suponha que o hash da mensagem seja $H(M) = 13$.

1. **Gerar vetor efêmero:** Bob cria um vetor aleatório "curto" $\mathbf{y}$, que será usado apenas para esta assinatura.

$$\mathbf{y} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

Parte 2: Geração da Assinatura

Bob quer assinar uma mensagem M. Suponha que o hash da mensagem seja $H(M) = 13$.

1. **Gerar vetor efêmero:** Bob cria um vetor aleatório "curto" $\mathbf{y}$, que será usado apenas para esta assinatura.

$$\mathbf{y} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

2. **Calcular o "compromisso":** Bob calcula $\mathbf{w} = \mathbf{A}\mathbf{y} \pmod{q}$.

$$\mathbf{w} = \begin{pmatrix} 10 & 3 \\ 5 & 9 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 16 \\ 23 \end{pmatrix} \equiv \begin{pmatrix} 16 \\ 6 \end{pmatrix} \pmod{17}$$

Parte 2: Geração da Assinatura

Bob quer assinar uma mensagem M. Suponha que o hash da mensagem seja $H(M) = 13$.

1. **Gerar vetor efêmero:** Bob cria um vetor aleatório "curto" $\mathbf{y}$, que será usado apenas para esta assinatura.

$$\mathbf{y} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

2. **Calcular o "compromisso":** Bob calcula $\mathbf{w} = \mathbf{A}\mathbf{y} \pmod{q}$.

$$\mathbf{w} = \begin{pmatrix} 10 & 3 \\ 5 & 9 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 16 \\ 23 \end{pmatrix} \equiv \begin{pmatrix} 16 \\ 6 \end{pmatrix} \pmod{17}$$

3. **Calcular o "desafio":** O desafio c é um hash da mensagem junto com o compromisso $\mathbf{w}$.

$$c = H(M \,||\, \mathbf{w}) = H(13\_16\_6) \rightarrow \text{Suponha que } c = 3$$

Parte 2: Geração da Assinatura

Bob quer assinar uma mensagem M. Suponha que o hash da mensagem seja $H(M) = 13$.

1. **Gerar vetor efêmero:** Bob cria um vetor aleatório "curto" $\mathbf{y}$, que será usado apenas para esta assinatura.

$$\mathbf{y} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

2. **Calcular o "compromisso":** Bob calcula $\mathbf{w} = \mathbf{A}\mathbf{y} \pmod{q}$.

$$\mathbf{w} = \begin{pmatrix} 10 & 3 \\ 5 & 9 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 16 \\ 23 \end{pmatrix} \equiv \begin{pmatrix} 16 \\ 6 \end{pmatrix} \pmod{17}$$

3. **Calcular o "desafio":** O desafio c é um hash da mensagem junto com o compromisso $\mathbf{w}$.

$$c = H(M \,||\, \mathbf{w}) = H(13\_16\_6) \rightarrow \text{Suponha que } c = 3$$

4. **Construir a assinatura:** Bob calcula o vetor $\mathbf{z} = \mathbf{y} + c \cdot \mathbf{s} \pmod{q}$.

$$\mathbf{z} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} + 3 \cdot \begin{pmatrix} 2 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} + \begin{pmatrix} 6 \\ -3 \end{pmatrix} = \begin{pmatrix} 7 \\ -1 \end{pmatrix} \equiv \begin{pmatrix} 7 \\ 16 \end{pmatrix} \pmod{17}$$

Conteúdo Público:

Mensagem: M

Chaves Pública: $(\mathbf{A}, \mathbf{t}) = (\begin{pmatrix} 10 & 3 \\ 5 & 9 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix})$

Assinatura Final: $(c, \mathbf{z}) = (3, \begin{pmatrix} 7 \\ 16 \end{pmatrix})$.

Parte 3: Verificação da Assinatura

Para validar, Alice precisa verificar se Bob realmente usou a chave secreta **s** dele. Ela faz isso recalculando o compromisso **w** e checando se o hash bate com o c recebido.

1. Alice calcula $\mathbf{w}' = \mathbf{A}\mathbf{z} - \mathbf{t}c \pmod{q}$.

Parte 3: Verificação da Assinatura

Para validar, Alice precisa verificar se Bob realmente usou a chave secreta **s** dele. Ela faz isso recalculando o compromisso **w** e checando se o hash bate com o c recebido.

1. Alice calcula $\mathbf{w}' = \mathbf{A}\mathbf{z} - \mathbf{t}c \pmod{q}$.
2. Primeiro, $\mathbf{A}\mathbf{z}$: $\mathbf{A}\mathbf{z} = \begin{pmatrix} 10 & 3 \\ 5 & 9 \end{pmatrix} \begin{pmatrix} 7 \\ 16 \end{pmatrix} = \begin{pmatrix} 118 \\ 179 \end{pmatrix} \equiv \begin{pmatrix} 16 \\ 9 \end{pmatrix} \pmod{17}$

Parte 3: Verificação da Assinatura

Para validar, Alice precisa verificar se Bob realmente usou a chave secreta **s** dele. Ela faz isso recalculando o compromisso **w** e checando se o hash bate com o c recebido.

1. Alice calcula $\mathbf{w}' = \mathbf{A}\mathbf{z} - \mathbf{t}c \pmod{q}$.
2. Primeiro, $\mathbf{A}\mathbf{z}$: $\mathbf{A}\mathbf{z} = \begin{pmatrix} 10 & 3 \\ 5 & 9 \end{pmatrix} \begin{pmatrix} 7 \\ 16 \end{pmatrix} = \begin{pmatrix} 118 \\ 179 \end{pmatrix} \equiv \begin{pmatrix} 16 \\ 9 \end{pmatrix} \pmod{17}$
3. Depois, $\mathbf{t}c$: $\mathbf{t}c = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \cdot 3 = \begin{pmatrix} 0 \\ 3 \end{pmatrix} \pmod{17}$

Parte 3: Verificação da Assinatura

Para validar, Alice precisa verificar se Bob realmente usou a chave secreta **s** dele. Ela faz isso recalculando o compromisso **w** e checando se o hash bate com o c recebido.

1. Alice calcula $\mathbf{w}' = \mathbf{A}\mathbf{z} - \mathbf{t}c \pmod{q}$.
2. Primeiro, $\mathbf{A}\mathbf{z}$: $\mathbf{A}\mathbf{z} = \begin{pmatrix} 10 & 3 \\ 5 & 9 \end{pmatrix} \begin{pmatrix} 7 \\ 16 \end{pmatrix} = \begin{pmatrix} 118 \\ 179 \end{pmatrix} \equiv \begin{pmatrix} 16 \\ 9 \end{pmatrix} \pmod{17}$
3. Depois, $\mathbf{t}c$: $\mathbf{t}c = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \cdot 3 = \begin{pmatrix} 0 \\ 3 \end{pmatrix} \pmod{17}$
4. Subtração para encontrar $\mathbf{w}'$: $\mathbf{w}' = \begin{pmatrix} 16 \\ 9 \end{pmatrix} - \begin{pmatrix} 0 \\ 3 \end{pmatrix} = \begin{pmatrix} 16 \\ 6 \end{pmatrix} \pmod{17}$

Parte 3: Verificação da Assinatura

Para validar, Alice precisa verificar se Bob realmente usou a chave secreta **s** dele. Ela faz isso recalculando o compromisso **w** e checando se o hash bate com o c recebido.

1. Alice calcula $\mathbf{w}' = \mathbf{A}\mathbf{z} - \mathbf{t}c \pmod{q}$.
2. Primeiro, $\mathbf{A}\mathbf{z}$: $\mathbf{A}\mathbf{z} = \begin{pmatrix} 10 & 3 \\ 5 & 9 \end{pmatrix} \begin{pmatrix} 7 \\ 16 \end{pmatrix} = \begin{pmatrix} 118 \\ 179 \end{pmatrix} \equiv \begin{pmatrix} 16 \\ 9 \end{pmatrix} \pmod{17}$
3. Depois, $\mathbf{t}c$: $\mathbf{t}c = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \cdot 3 = \begin{pmatrix} 0 \\ 3 \end{pmatrix} \pmod{17}$
4. Subtração para encontrar $\mathbf{w}'$: $\mathbf{w}' = \begin{pmatrix} 16 \\ 9 \end{pmatrix} - \begin{pmatrix} 0 \\ 3 \end{pmatrix} = \begin{pmatrix} 16 \\ 6 \end{pmatrix} \pmod{17}$
5. **Verificação final:** Alice verifica se c é igual ao *hash* da mensagem com o $\mathbf{w}'$ que ela acabou de calcular.

$$c \stackrel{?}{=} H(M \,||\, \mathbf{w}') \rightarrow H(13\_16\_6) \rightarrow 3$$

Parte 3: Verificação da Assinatura

Para validar, Alice precisa verificar se Bob realmente usou a chave secreta **s** dele. Ela faz isso recalculando o compromisso **w** e checando se o hash bate com o c recebido.

1. Alice calcula $\mathbf{w}' = \mathbf{A}\mathbf{z} - \mathbf{t}c \pmod{q}$.
2. Primeiro, $\mathbf{A}\mathbf{z}$: $\mathbf{A}\mathbf{z} = \begin{pmatrix} 10 & 3 \\ 5 & 9 \end{pmatrix} \begin{pmatrix} 7 \\ 16 \end{pmatrix} = \begin{pmatrix} 118 \\ 179 \end{pmatrix} \equiv \begin{pmatrix} 16 \\ 9 \end{pmatrix} \pmod{17}$
3. Depois, $\mathbf{t}c$: $\mathbf{t}c = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \cdot 3 = \begin{pmatrix} 0 \\ 3 \end{pmatrix} \pmod{17}$
4. Subtração para encontrar $\mathbf{w}'$: $\mathbf{w}' = \begin{pmatrix} 16 \\ 9 \end{pmatrix} - \begin{pmatrix} 0 \\ 3 \end{pmatrix} = \begin{pmatrix} 16 \\ 6 \end{pmatrix} \pmod{17}$
5. **Verificação final:** Alice verifica se c é igual ao *hash* da mensagem com o $\mathbf{w}'$ que ela acabou de calcular.

$$c \stackrel{?}{=} H(M \,||\, \mathbf{w}') \rightarrow H(13\_16\_6) \rightarrow 3$$

Parte 3: Verificação da Assinatura

Para validar, Alice precisa verificar se Bob realmente usou a chave secreta **s** dele. Ela faz isso recalculando o compromisso **w** e checando se o hash bate com o c recebido.

1. Alice calcula $\mathbf{w}' = \mathbf{A}\mathbf{z} - \mathbf{t}c \pmod{q}$.
2. Primeiro, $\mathbf{A}\mathbf{z}$: $\mathbf{A}\mathbf{z} = \begin{pmatrix} 10 & 3 \\ 5 & 9 \end{pmatrix} \begin{pmatrix} 7 \\ 16 \end{pmatrix} = \begin{pmatrix} 118 \\ 179 \end{pmatrix} \equiv \begin{pmatrix} 16 \\ 9 \end{pmatrix} \pmod{17}$
3. Depois, $\mathbf{t}c$: $\mathbf{t}c = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \cdot 3 = \begin{pmatrix} 0 \\ 3 \end{pmatrix} \pmod{17}$
4. Subtração para encontrar $\mathbf{w}'$: $\mathbf{w}' = \begin{pmatrix} 16 \\ 9 \end{pmatrix} - \begin{pmatrix} 0 \\ 3 \end{pmatrix} = \begin{pmatrix} 16 \\ 6 \end{pmatrix} \pmod{17}$
5. **Verificação final:** Alice verifica se c é igual ao *hash* da mensagem com o $\mathbf{w}'$ que ela acabou de calcular.

$$c \stackrel{?}{=} H(M \,||\, \mathbf{w}') \rightarrow H(13\_16\_6) \rightarrow 3$$

Assinatura Válida!

Do Exemplo à Realidade: O Bloco de Construção

No nosso exemplo, usamos inteiros simples. Na realidade, o ML-DSA opera sobre **polinômios**.

- Um polinômio no ML-DSA tem 256 coeficientes:

$$p(X) = a_{255}X^{255} + a_{254}X^{254} + \cdots + a_1X + a_0$$

- Cada coeficiente a_i é um número inteiro módulo q, onde q é um primo grande de 23 bits ($q = 8380417$).

Do Exemplo à Realidade: O Bloco de Construção

No nosso exemplo, usamos inteiros simples. Na realidade, o ML-DSA opera sobre **polinômios**.

- Um polinômio no ML-DSA tem 256 coeficientes:

$$p(X) = a_{255}X^{255} + a_{254}X^{254} + \cdots + a_1X + a_0$$

- Cada coeficiente a_i é um número inteiro módulo q, onde q é um primo grande de 23 bits ($q = 8380417$).

Como armazenar um polinômio em um computador?

- De forma muito simples: como uma lista (ou array) de seus 256 coeficientes.

$$p(X) \quad \longrightarrow \quad [a_0, a_1, a_2, \ldots, a_{255}]$$

Ideia Central

Todas as estruturas do ML-DSA (vetores e matrizes) são, na verdade, vetores e matrizes **de polinômios**.

Juntando as Peças: Matrizes de Polinômios

Vamos usar o **ML-DSA-2** (Nível de Segurança 2 do NIST) como exemplo concreto.

- **A Matriz Pública A:**
  - No nosso exemplo, **A** era uma matriz 2x2 de inteiros.
  - No ML-DSA-2, **A** é uma matriz **4x4 de polinômios**.
  - **Tamanho total de A:** $4 \times 4 = 16$ polinômios. Cada um com 256 coeficientes. São $16 \times 256 = 4096$ coeficientes no total!

- **Chaves e Assinatura:**
  - A chave privada **s** e a pública **t** são **vetores de polinômios**.
  - A assinatura **z** também é um vetor de polinômios.

Aspectos Práticos: O Tamanho Real

Compressão é a Chave!

Os tamanhos finais em bytes são menores do que a matemática bruta sugere. O padrão usa técnicas de compressão inteligentes (descartando bits menos significativos) para reduzir o tamanho da chave pública e da assinatura.

Tabela: Tamanhos Finais em Bytes (ML-DSA-2)

	ECDSA (P-256)	**ML-DSA-2**	**Diferença**
Tam. Chave Pública	64 bytes	1312 bytes	**20x Maior**
Tam. da Assinatura	64 bytes	2420 bytes	**37x Maior**

Índice

Exercício: Serviços de Segurança

1) Neste problema, consideramos alguns aspectos básicos dos serviços de segurança. Justifique todas as suas respostas.

1. Afirma-se que a autenticação do remetente (ou da mensagem) sempre implica em integridade dos dados. Por quê? A recíproca é verdadeira, ou seja, a integridade dos dados sempre implica em autenticação do remetente?

2. A confidencialidade sempre garante a integridade?

Solução: Serviços de Segurança

1. **Autenticação implica em Integridade, mas o contrário não é verdadeiro.**
   - **Por que sim?** A autenticação cria um "selo" criptográfico (MAC ou assinatura) que depende da chave secreta do remetente **e** do conteúdo exato da mensagem. Se a mensagem for alterada, o selo se quebra e a verificação falha, garantindo assim a integridade.
   - **Por que não o contrário?** Um mecanismo de integridade puro (como um Checksum/CRC) não usa segredos. Qualquer um, incluindo um invasor, pode criar um checksum válido para uma mensagem adulterada. Portanto, ele não prova quem enviou a mensagem.
2. **Não, confidencialidade não garante integridade.**
   - A confidencialidade esconde o conteúdo da mensagem, mas não impede que ela seja alterada. Um invasor pode interceptar um texto cifrado e modificar alguns de seus bits. O destinatário, ao decifrar, obterá um texto simples corrompido e sem sentido, mas não terá um mecanismo para detectar que a alteração ocorreu. Alguns modos de operação de cifras de bloco (como o CBC) são vulneráveis a ataques de "bit-flipping" que exploram exatamente isso.

Exercício: Esquema de Assinatura RSA

2) Neste problema, analisamos o esquema de assinatura RSA.

1. Calcule a assinatura RSA para a mensagem $x = 1234$ e o módulo $n = 11111$. Escolha o menor valor possível para o expoente e.

2. Qual é o benefício de escolher um expoente público e pequeno? Justifique sua resposta!

Solução: Esquema de Assinatura RSA

Cálculo da Assinatura RSA:

- **Passo 1: Fatorar n e calcular $\phi(n)$.**
  $n = 11111 = 41 \cdot 271$. Ambos são primos.
  $\phi(n) = (p-1)(q-1) = (40)(270) = 10800$.
- **Passo 2: Encontrar o menor expoente e.**
  e deve ser um primo relativo a $\phi(n) = 10800$. O menor inteiro $e > 1$ que não divide 10800 é $e = 7$.
- **Passo 3: Calcular a chave privada d.**
  $d \equiv e^{-1} \pmod{\phi(n)} \rightarrow d \equiv 7^{-1} \pmod{10800}$.
  Usando o Algoritmo Estendido de Euclides, encontramos $d = 1543$.
- **Passo 4: Assinar a mensagem.**
  A assinatura s é calculada como $s \equiv x^d \pmod{n}$.
  $s \equiv 1234^{1543} \pmod{11111}$.
  Calculando a exponenciação modular, obtemos $s = 8182$.

Benefício de um expoente e pequeno:

- A principal vantagem é a **eficiência na verificação da assinatura**. A verificação requer o cálculo de $s^e \pmod{n}$. Se e for um número pequeno (como 3, 7 ou 65537), esta exponenciação é extremamente rápida, pois envolve poucas multiplicações. Isso é ideal para dispositivos com pouca capacidade de processamento (como smart cards) ou para servidores que precisam verificar milhares de assinaturas por segundo.

Exercício: Verificação de Assinaturas RSA

3) Dado um esquema de assinatura RSA com a chave pública $(n = 9797, e = 131)$, quais das seguintes assinaturas são válidas?

- 1. $(x = 123, \text{sig}(x) = 6292)$
- 2. $(x = 4333, \text{sig}(x) = 4768)$
- 3. $(x = 4333, \text{sig}(x) = 1424)$